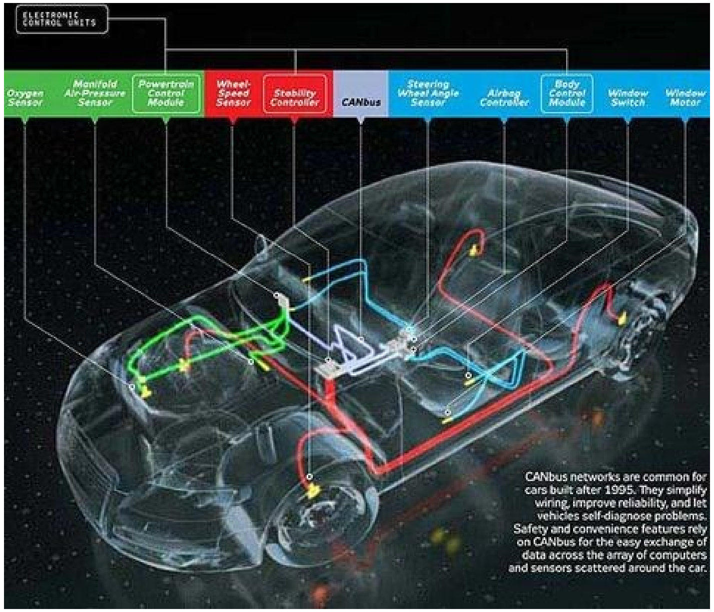
Computer Abstractions and Technology

Prof. Yongtae Kim

Computer Science and Engineering
Kyungpook National University

Introduction

- **The computer revolution continues and the advances in computer now affect almost every aspect of our society**
  - Computer in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search engines



Classes of Computers

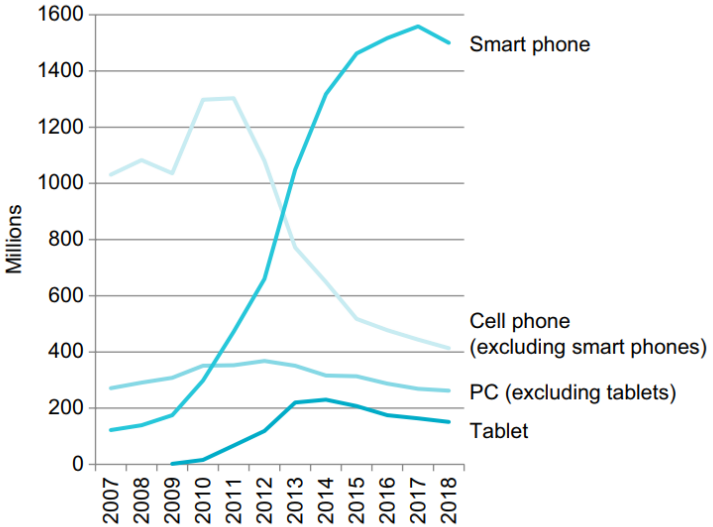
- **Personal computers (PCs)**
  - General purpose and variety of software
  - Subject to cost and performance tradeoff
  - e.g.) desktop, laptop

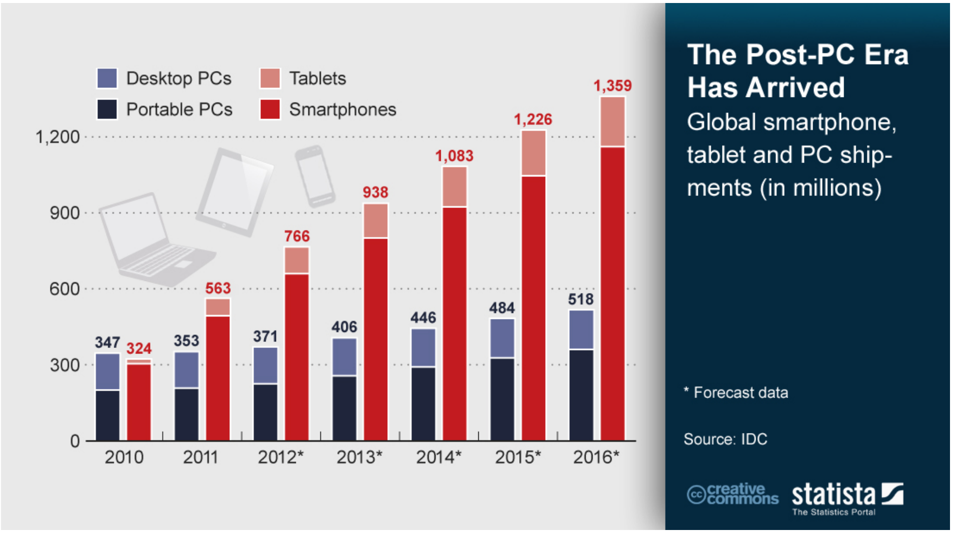
- **Servers**
  - Usually accessed via network
  - High capacity, performance, and reliability
  - Range from small servers to building sized
    - Low-end: Used for small business or web service
    - High-end: Supercomputers or datacenter with hundreds to thousands processors with tera bytes of memory and petabytes of storage

- **Embedded computers**
  - A computer inside another device used for running predetermined applications (e.g., smartphones, tablet)
  - Stringent power, performance, and cost constraints

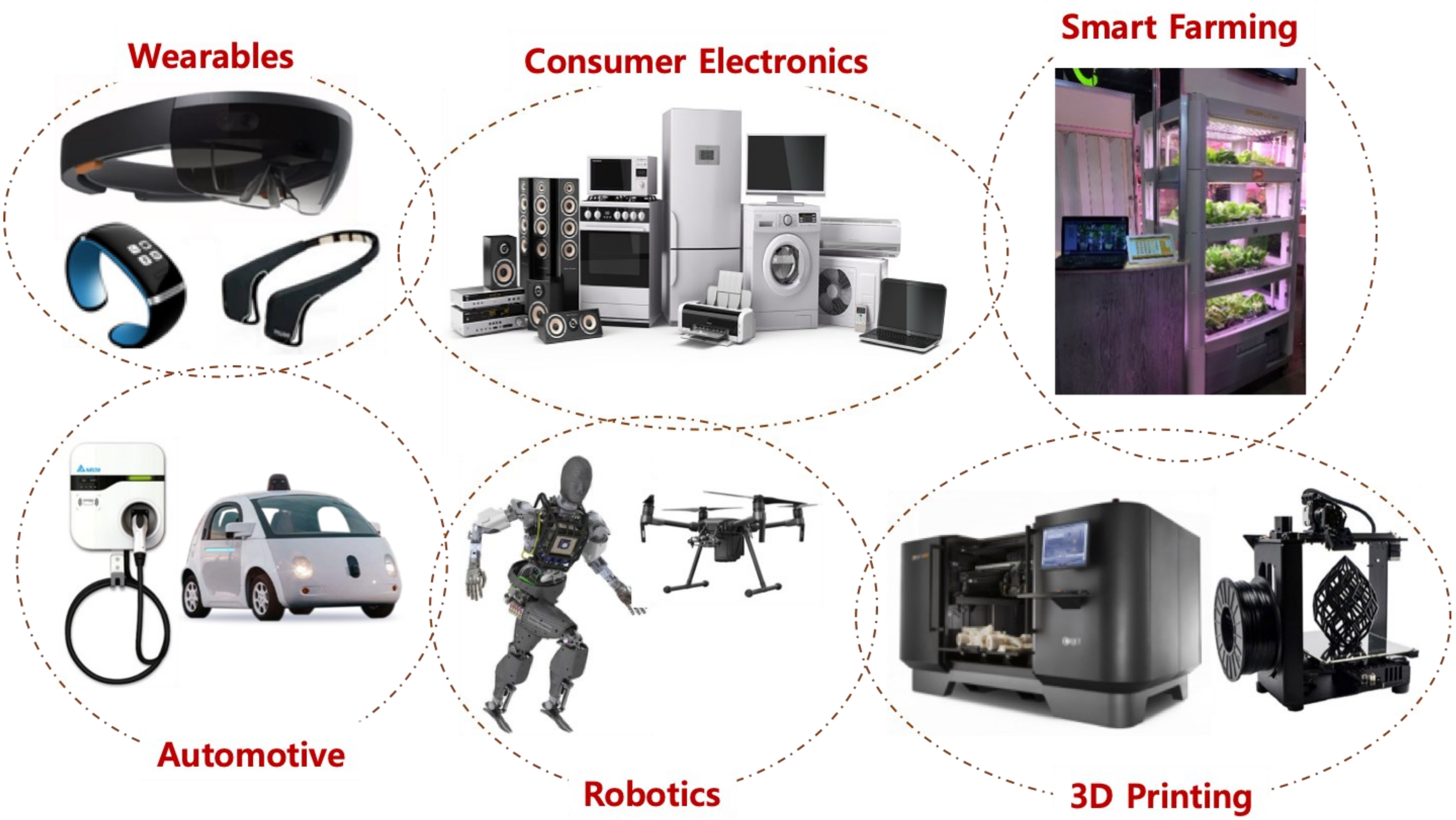
Post-PC Era

- **Replacing PC is the personal mobile devices (PMDs)**
  - PMDs are battery operated with wireless connectivity to the Internet
  - Typically cost hundreds of dollars, no keyboard and mouse
  - e.g.) smart phones, tablets, and electronic glasses
- **Taking over from the conventional server is cloud computing**
  - Kwon as warehouse scale computers (WSCs)
  - Software as a Service (SaaS) is deployed via cloud computing
  - e.g.) Amazon and google





Variety of Post-PCs



Automotive Electronics

Powertrain Management

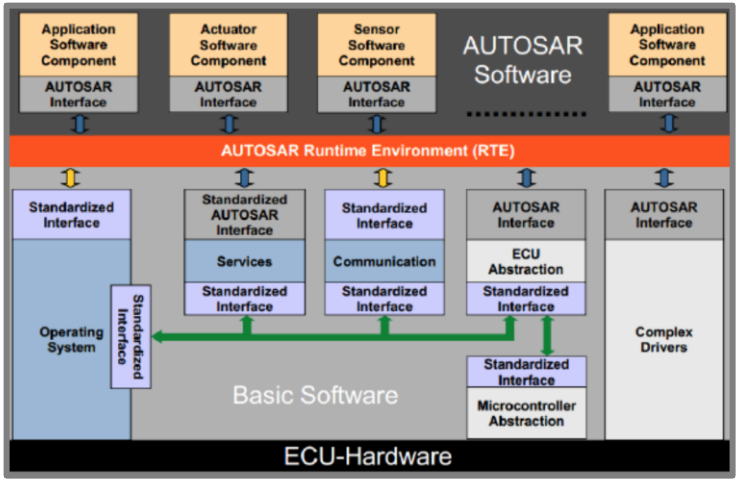
Advanced Driver Assistance System (ADAS)

Body Controllers

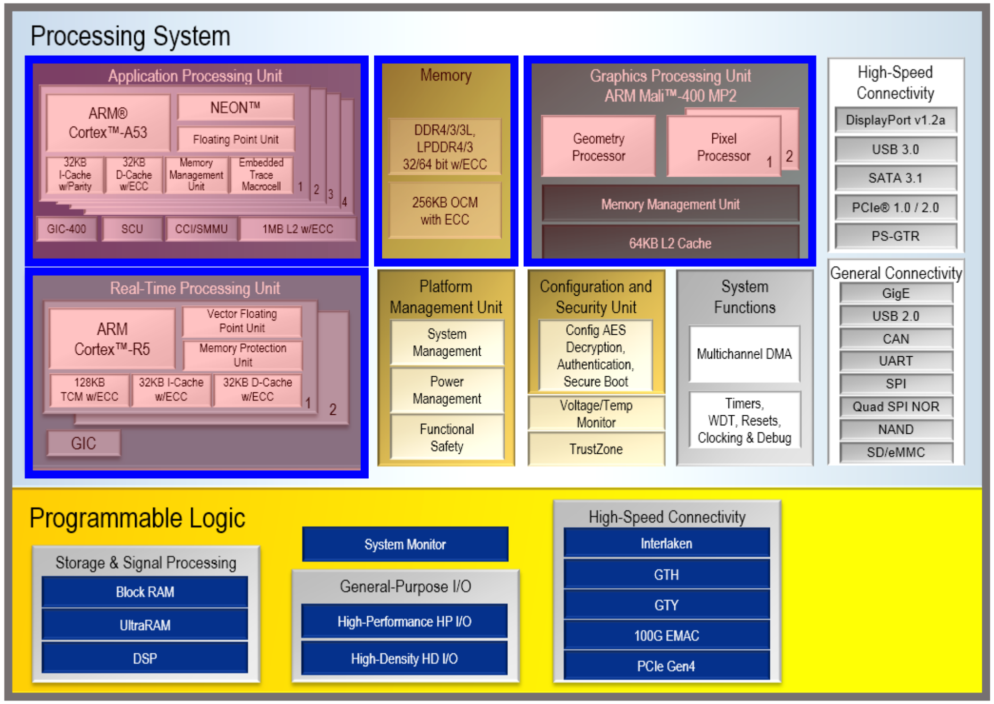
Infotainment System

Electronic Control Unit (ECU)

- **ECU is an embedded system in automotive electronics that controls on or more of the electrical systems in a vehicle**
  - If you want to handle automotive electronics using the software, we need to start learning computer architecture first!



AUTOSAR Platform Architecture



Xilinx Zynq UltraScale + MPSoC

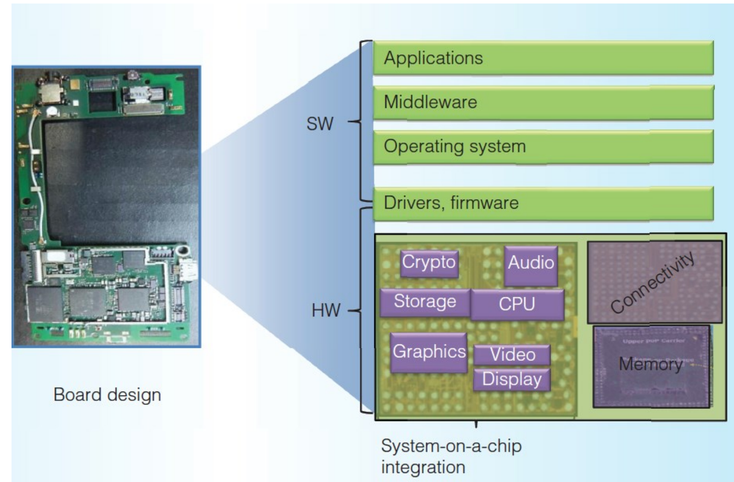
What You Will Learn in this Class

- **How programs are translated into the machine language**
  - How the hardware executes them
- **The hardware and software interface**
  - Instruction set architecture (ISA)
- **What determines program performance**
  - How it can be improved
- **How hardware designers improve performance**
- **Understanding program performance**

Hardware or software component	How this component affects performance
Algorithm	Determines both the number of source-level statements and the number of I/O operations executed
Programming language, compiler, and architecture	Determines the number of computer instructions for each source-level statement
Processor and memory system	Determines how fast instructions can be executed
I/O system (hardware and operating system)	Determines how fast I/O operations may be executed

What is Computer Architecture?

- **Computer Architecture = ISA + Machine Organization**



Application

OS

Compiler | Firmware

Instruction Set Architecture

Processor | I/O system

Logic design

Circuit design

Physical design (Layout)

Machine Organization

Computer Architecture

Abstraction Analogies

Driver

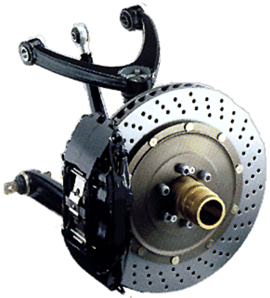
Customer

Abstraction Layer

Abstraction Layer

Machine Details

Combustion Engine in a car



Brake system in a car

Machine Details

Hardware board in a vending machine

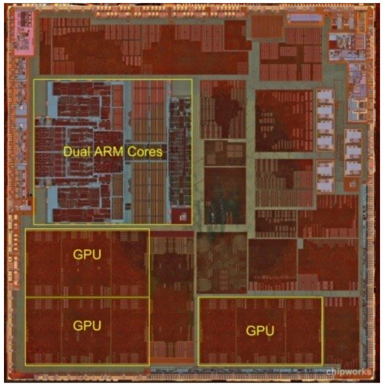
Abstractions in Smartphones

Application Programming using APIs

Operating Systems

Instruction Set Architecture (ISA)

Hardware Implementation (ARM)



Eight Great Ideas in Computer Architecture

- **Design for Moore's Law**
  - Integrated circuit resources double every 18 ~ 24 months
- **Use abstraction to simplify design**
  - Abstractions to characterize the design at difference levels
- **Make the common case fast**
  - Enhance performance better than optimizing the rare case
- **Performance via parallelism**
  - Get more performance by computing operations in parallel
- **Performance via pipelining**
  - A particular pattern of parallelism
- **Performance via prediction**
  - Faster on average to guess and start working rather than waiting
- **Hierarchy of memories**
  - Address size and speed conflict with a hierarchy of memories
- **Dependability via redundancy**
  - Make system dependable by including redundant components


MOORE'S LAW


ABSTRACTION


COMMON CASE FAST


PARALLELISM


PIPELINING

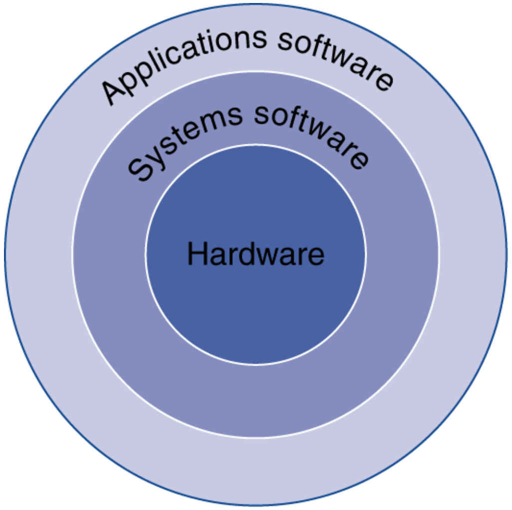

PREDICTION


HIERARCHY


DEPENDABILITY

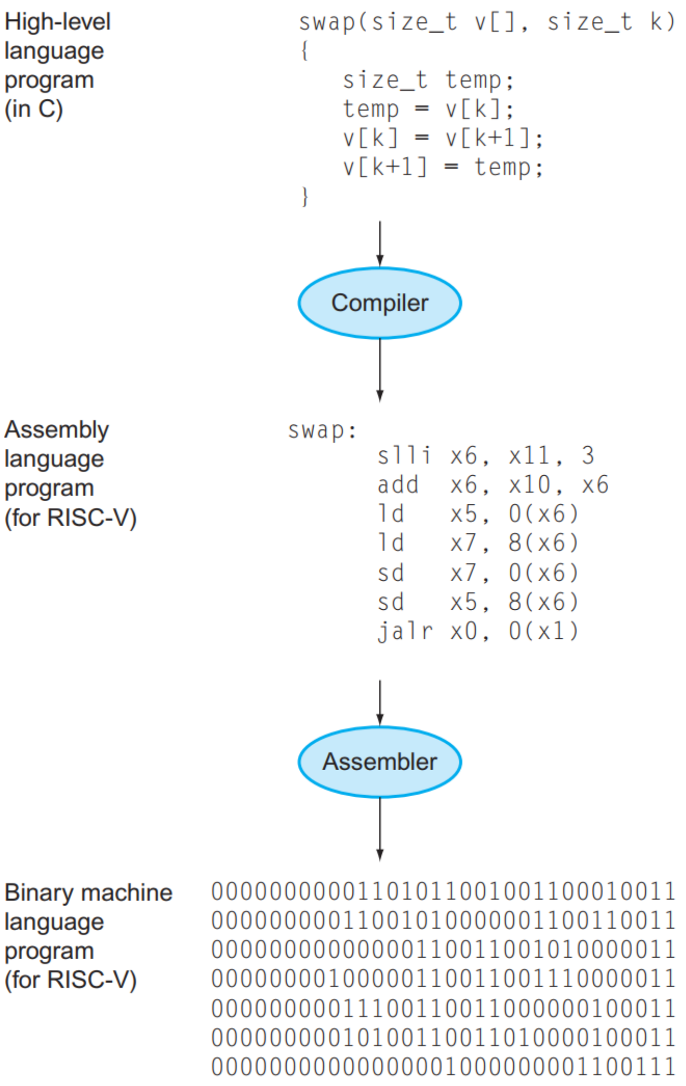
Below Your Program

- The layers of software are organized primarily in a hierarchical fashion
- **Application software** is written in high-level language
- **System software** is sitting between the hardware and application software
  - Operating system interfaces between a user's program and the hardware
    - Handling basic input and output operation
    - Allocating storage and memory
    - Resource sharing and task scheduling
    - e.g.) Linux, Windows, iOS
  - Compiler translates of a program written in high-level languages into instructions that the hardware executes
- **Hardware** includes processor, memory, I/O controller, etc



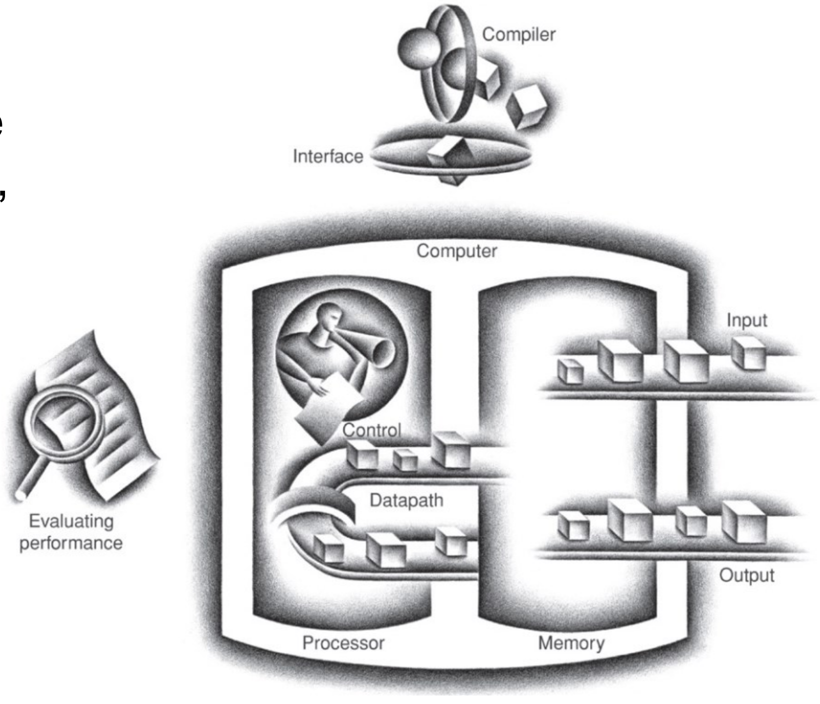
High-Level Language to Language of Hardware

- **High-level language is a level of abstraction close to problem domain**
  - Providing for productivity and portability (e.g., C++)
  - Compilers translate programs into instructions
- **Assembly language is a symbolic representation of machine instructions**
  - Assemblers translate a symbolic version of instructions into the binary version
- **Machine language is a hardware representation of machine instructions**
  - Represented by binary digits (i.e., bits)
  - Encoded instructions and data



Organization of a Computer

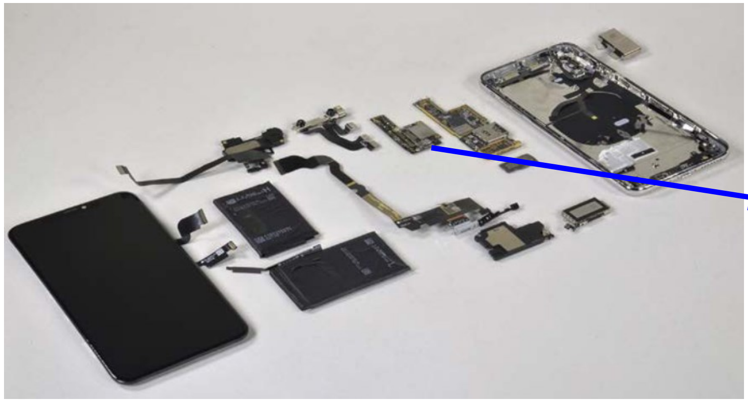
- **The five classic components of a computer are input, output, memory, datapath, and control**
  - Processor: datapath + control
  - The processor gets instructions and data from memory
  - Input writes data to memory
  - Output reads data from memory
  - Control sends the signals that determine the operations of the data path, memory, input, and output



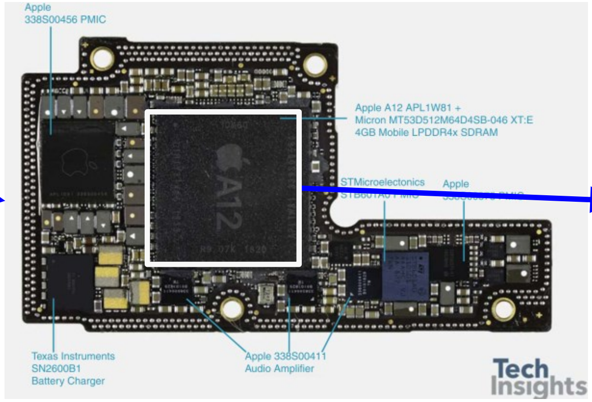
- **Same components for all kinds of computer**
  - e.g.) desktop, server, and embedded

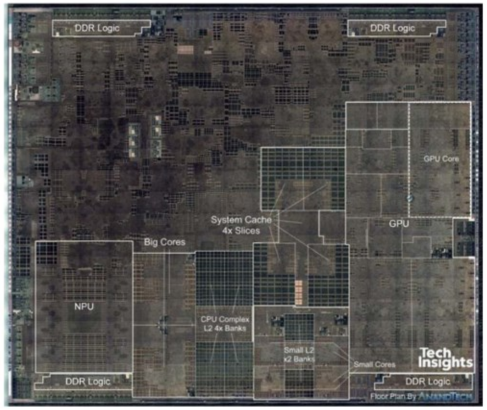
Opening the Box (1)

- **I/O of the five components dominate the Apple iPhone XS Max**
  - I/O: a capacitive multitouch LCD, front-/rear-facing cameras, microphone, speakers, Wi-Fi and Bluetooth networks, etc
  - The datapath, control, and memory are a tiny portion
- **There are many integrated circuits (i.e., chips) in the board**
  - A12 processor contains two large and four little ARM processors (aka CPU: Central Processing Unit) that operate with a clock rate of 2.5 GHz
    - Datapath performs arithmetic operations
    - Control sequences datapath, memory, and I/O



Apple iPhone XS Max





Opening the Box (2)

- **iPhone XS Max package includes a memory chip with 32 Gb**
  - Memory is the storage area where programs and data are kept when running
  - The memory is built from dynamic random access memory (DRAM) chips that contain the instructions and data of a program
- **Cache memory consists of a small, fast memory that acts as a buffer for the DRAM**
  - Cache is built using static random access memory (SRAM), which is faster but less dense, and hence more expensive, than DRAM
  - SRAM and DRAM are two layers of the memory hierarchy
- **Abstraction helps us deal with complexity by hiding lower-level details**
  - Abstract of interface between the hardware and the lowest-level software is instruction set architecture (ISA), or simply architecture
  - Application binary interface (ABI) is the ISA + OS interface
  - Implementation is the hardware that obeys architecture abstraction

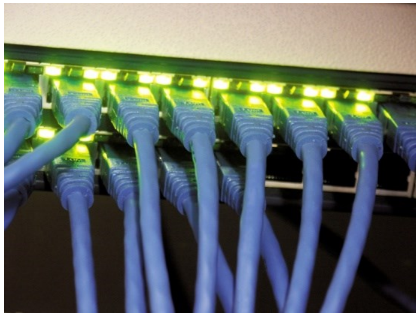
A Safe Place for Data

- **Volatile memory loses instructions and data when power off**
  - Main memory or primary memory
  - Usually small size and fast
  - e.g.) DRAM and SRAM

- **Nonvolatile memory keeps programs and data when power off**
  - Secondary memory
  - Larger size and slower than the main memory
  - e.g.) magnetic (hard) disk, flash memory, optical disk (CD, DVD)

Communicating with Other Computers

- **Networks interconnect whole computers and networked computers have several major advantages**
  - Communication: Information is exchanged between computers at high speeds
  - Resource sharing: Computers on the network can share I/O devices
  - Nonlocal access: Users need NOT be near the computer they are using



- **Local area network (LAN) is designed to carry data within a geographically confined area, typically within a single building: Ethernet**
- **Wide area network (WAN) is a network extended over hundreds of kilometers that can span a continent: Internet**
- **Wireless network: Wi-Fi, Bluetooth**

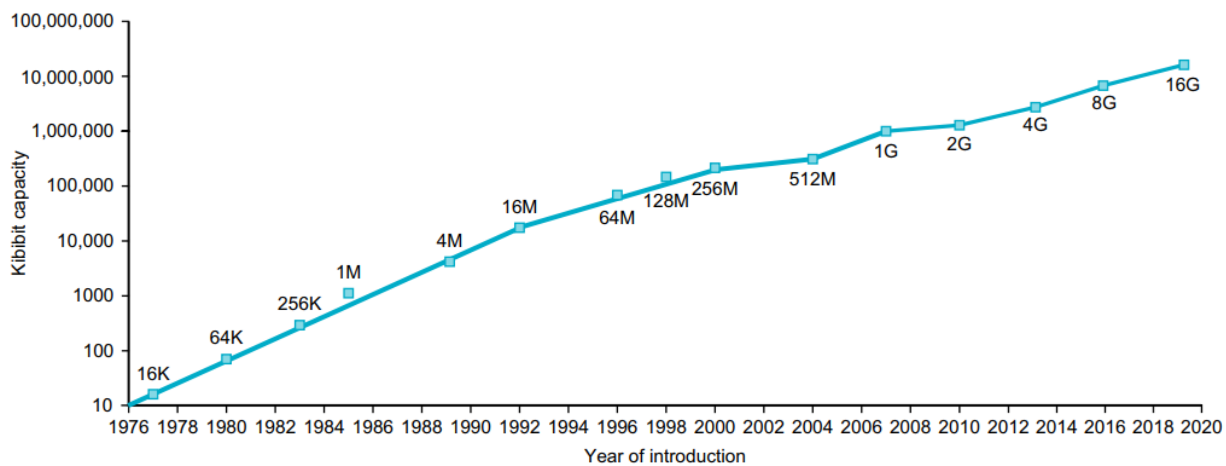
Technology Trends

- **Processors and memory have improved at an incredible rate**

Year	Technology used in computers	Relative performance/unit cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit	900
1995	Very large-scale integrated circuit	2,400,000
2020	Ultra large-scale integrated circuit	500,000,000,000

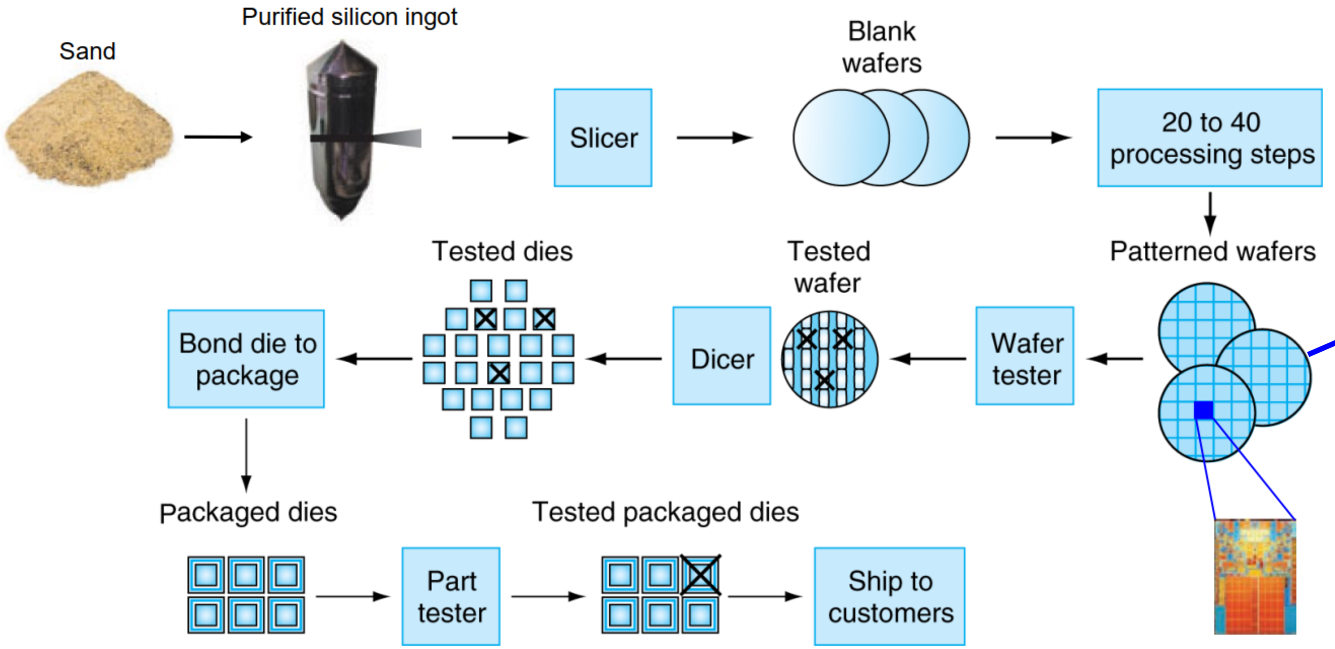
- **A transistor is simply an on/off switch controlled by electricity**
  - Integrated circuit (IC) combined dozens to hundreds of transistors into a single chip
  - Very large scale integrated circuit (VLSI) contains hundreds to millions of transistors
- **The rate of increasing integration has been remarkably stable**

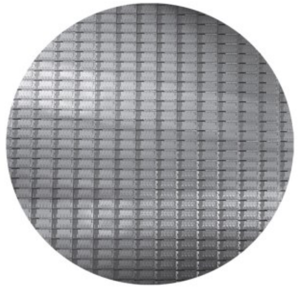
DRAM capacity



Chip Manufacturing Process

- **The manufacture of a chip begins with silicon, a substance found in sand**
  - Silicon is called semiconductor because it does NOT conduct electricity well
  - With a special chemical process, it can be 1) excellent conductors, 2) excellent insulator, 3) areas that can conduct or insulate under specific conditions (switch)

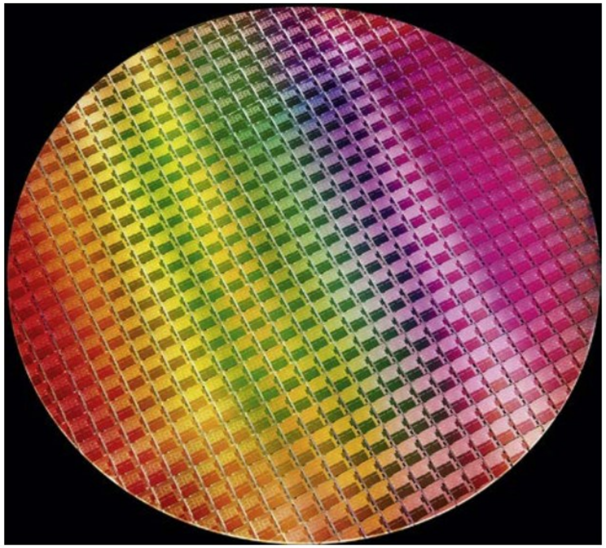




12 inch wafer of Intel Core i7

Cost of Integrated Circuit

- **10th Gen Intel® Core™ Processor (Ice Lake) Wafer**
  - 12 inch (300 mm) wafer, 506 chips (100% yield), 10-nm technology
  - Each chip is $11.4 \times 10.7\ \text{mm}^2$



- **Cost of a chip**

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area}/\text{Die area}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

- **Nonlinear relation to area and defect rate**
  - Wafer cost and area are fixed
  - Defect rate determined by manufacturing process
  - Die area determined by architecture and circuit design

Defining Performance

- **Which airplane has the best performance?**

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 737	240	3000	564	135,360
BAC/Sud Concorde	132	4000	1350	178,200
Boeing 777-200LR	301	9395	554	166,761
Airbus A380-800	853	8477	587	500,711

- **Performance considerations for computer**
  - Processor characteristics (e.g., clock frequency, # of cores)
  - System bus speed
  - Cache / Memory size
  - Disk storage (SSD or HDD)
  - Graphic cards (GPU)
  - Power consumption
  - Price, weight, form factor, etc

Response Time and Throughput

- **Individual computer user is interested in reducing response time**
  - The time between the start and completion of a task, called execution time
- **Datacenter manager cares about increasing throughput (bandwidth)**
  - The total amount of work done in a given time
- **How are response time and throughput affected by**
  - Replacing the processor with a faster one → both response time and throughput ↑
  - Adding additional processors → only throughput ↑
- **Our primary concern is the response time and the performance of a computer X can be defined by**

$$Performance_X = \frac{1}{Execution\ time_X}$$

- **"X is n times faster than Y"**

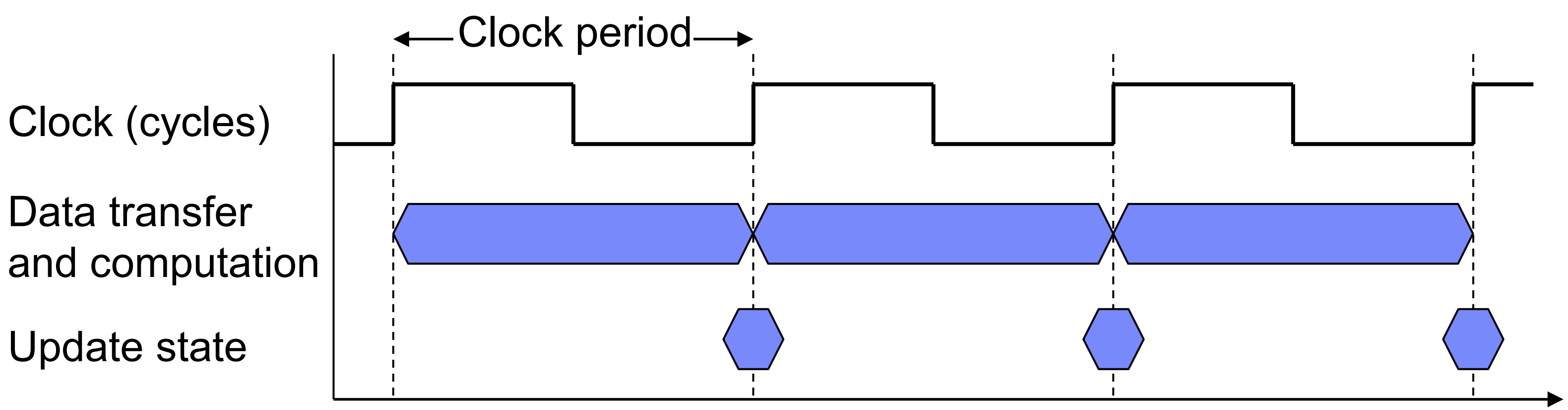
$$\frac{Performance_X}{Performance_Y} = \frac{Execution\ time_Y}{Execution\ time_X} = n$$

Measuring Performance

- **Time is the measure of computer performance**
  - Program execution time is measured in seconds per program
- **The most straightforward definition of time is called wall clock time, response time, or elapsed time**
  - These terms mean the total time to complete a task, including disk/memory accesses, input/output (I/O) activities, operating system overheads – everything
- **CPU execution time (or CPU time) is the time the CPU spends computing for the task**
  - This does NOT include the time consumed by I/O or other programs
  - CPU time can be further divided into user CPU time and system CPU time
- **Different applications are sensitive to different aspects of the performance of a computer system**
  - To improve the performance of a program, one must have a clear definition of what performance metric matters
  - Then, proceed to find performance bottlenecks by measuring program execution

CPU Clocking

- **Almost all computers are constructed using a clock that determines when events take place in the hardware**
  - These discrete time intervals are called clock cycles (or ticks, clock ticks, clock periods, clocks, cycles)



- **Designers refer to the length of a clock period both as the time for a complete clock cycle and as the clock rate (frequency)**
  - Clock period: duration of a clock cycle (e.g., 250 ps)
  - Clock rate (or clock frequency): cycles per second (e.g., 4 GHz)

CPU Performance

- **The bottom-line performance measure is CPU execution time**

$$\text{CPU execution time} = \text{CPU clock cycles} \times \text{Clock cycle time}$$
$$= \frac{\text{CPU clock cycles}}{\text{Clock rate}}$$

- **The hardware designer can improve performance by**
  - Reducing the number of clock cycles required for a program
  - Reducing the length of the clock cycle (i.e., increasing the clock rate)
- **The designer often faces a trade-off between the number of clock cycles and the clock rate**
- **Please see the example at pp. 34 ~ 35**

Instruction Performance

- **The execution time depends on the number of instructions in a program**

$$\text{Clock cycles} = \text{Instruction count} \times \text{Cycles per instruction (CPI)}$$

$$\begin{aligned}\text{CPU execution time} &= \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time} \\ &= \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}\end{aligned}$$

- **Instruction count for a program is determined by program, ISA, and compiler**
- **CPI (Clock cycle per instruction) is the average number of clock cycles each instruction take to execute for a program**
  - Determined by CPU hardware
  - Different instructions may take different clock cycles → CPI is an average
- **Please see the examples at pp. 35 ~ 37**

Performance Summary

- **All things considered, the execution time measured in seconds per program:**

$$\text{CPU Time} = \text{Seconds} / \text{Program} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

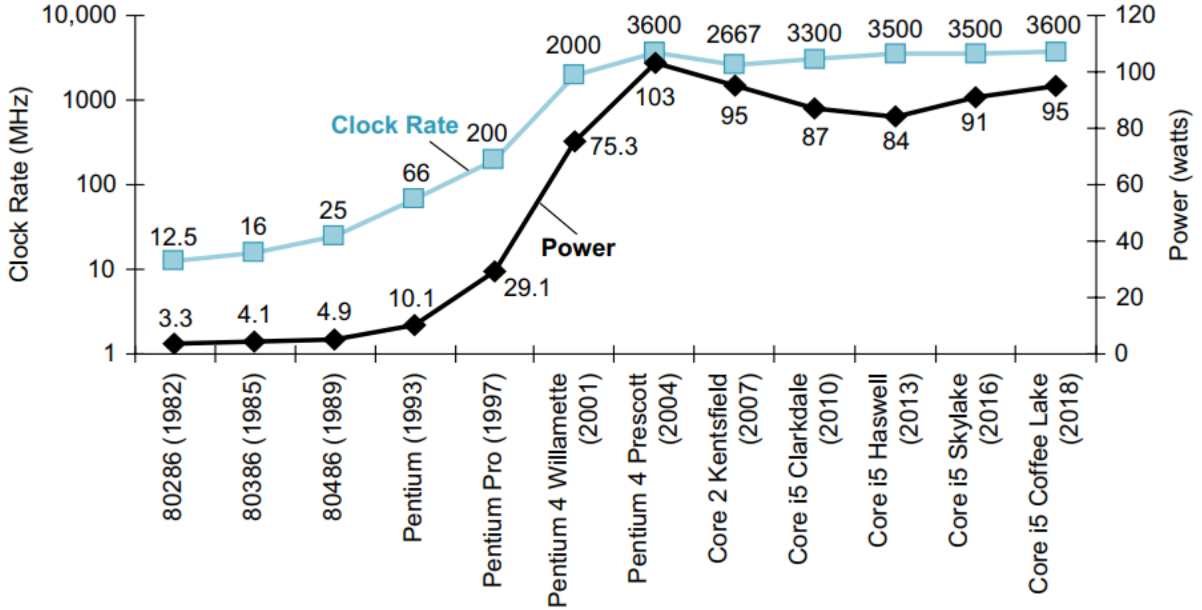
Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

- **The performance of a program depends on:**

Component	Instruction count	CPI	Clock frequency
Algorithm	V	V	
Programming Language	V	V	
Compiler	V	V	
Instruction Set Architecture	V	V	V
Technology			V

Power Trend

- **Both clock rate and power increased rapidly for decades and then flattened or dropped off recently**



- **In the post-PC era, the really valuable resource is energy**
  - In CMOS (complementary metal oxide semiconductor) technology, which is the dominant technology for IC, the dynamic energy is

$$\text{Energy} \propto \text{Capacitive load} \times \text{Voltage}^2$$

Power Wall

- **The power is the product of the energy and the frequency**

$$\text{Power} \propto \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

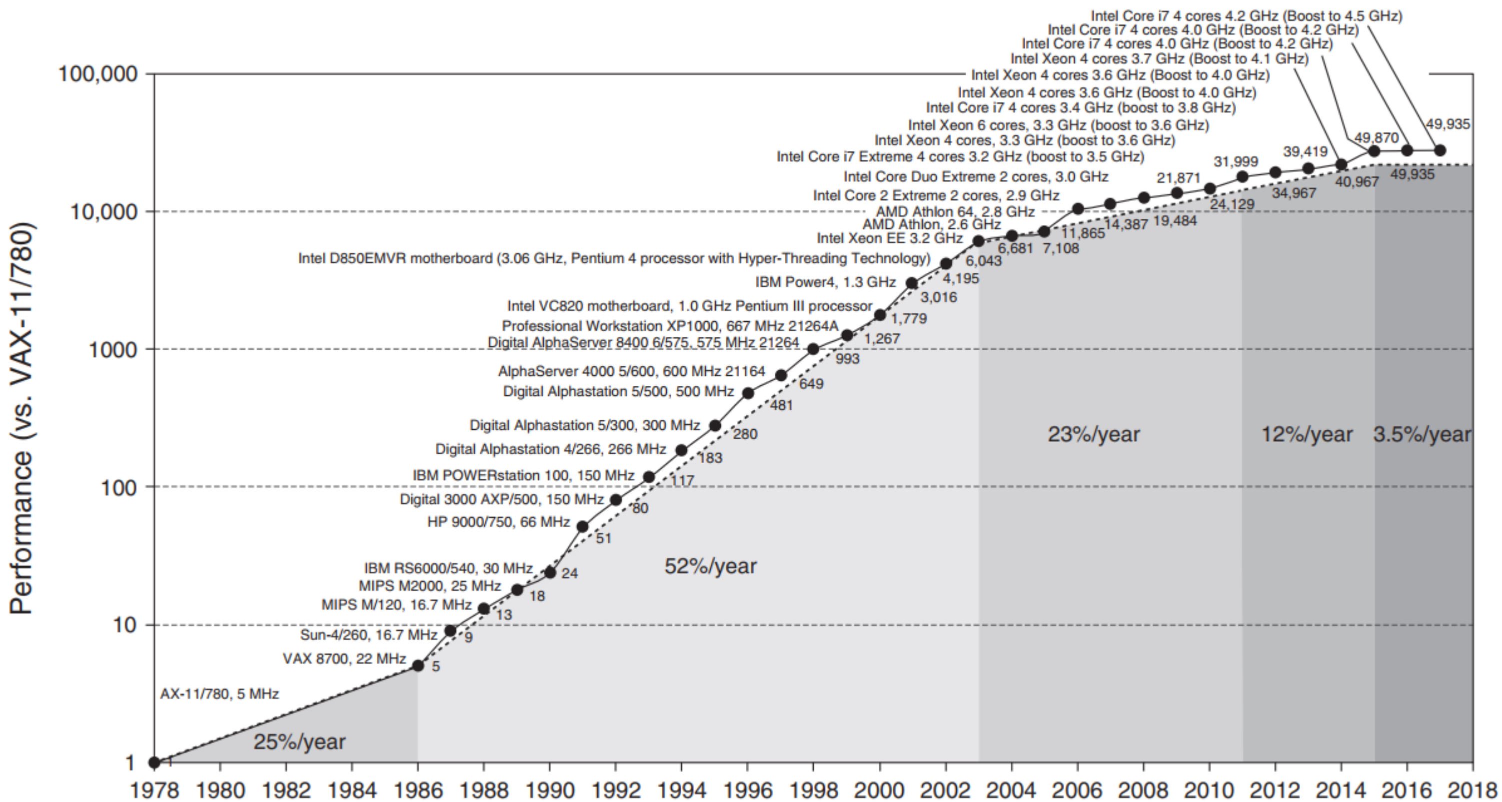
- **How could clock rates grow by a factor of 1000 while power increased by only a factor of 30 in the last 20 years?**
  - Voltage have gone from 5V to 1V, which is why the increase in power is only 30 times
  - Typically, the voltage was reduced about 15% per generation

- **The power wall is that**
  - We can NOT reduce the voltage further → leakage
  - We can NOT remove more heat → cooling cost

- **Please see the examples at pp. 41 ~ 42**

Switch from Uniprocessors to Multiprocessors

- **The power limit has forced a dramatic change in the design of microprocessors**
  - As of 2006, all desktop and server companies are shipping microprocessors with multiple cores per chip, where benefit is more on throughput than on response time



Amdahl's Law

- **Improving an aspect of a computer and expecting a proportional improvement in overall performance**

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- **Suppose a program runs in 100 seconds on a computer, with multiply operations responsible for 80 seconds of this time**
  - How much improvement in multiply performance to get 5× overall?

$$20\text{seconds} = \frac{80\text{seconds}}{n} + (100 - 80\text{seconds})$$

$$= \frac{80\text{seconds}}{n} + 20\text{seconds}$$

$$0 = \frac{80\text{seconds}}{n}$$

- **Therefore, we need to make the common case fast to improve the overall performance**

MIPS as a Performance Metric

- **MIPS (million instructions per second) is given by**

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

  – Faster computers have a higher MIPS rating

- **CPI varies between programs on a given CPU**

- **MIPS does NOT account for**
  – differences in ISAs between computers
  – differences in complexity between instructions